# CJEVOVODI I IMENOVANI CJEVOVODI

## CJEVOVODI

Općenito se procesi povezuju cjevovodom na slijedeći način:

1. Napraviti cjevovod.
2. Napraviti proces dijete koji će čitati.
3. U djetetu zatvoriti kraj cjevovoda na kojeg se piše i obaviti druge pripreme ako je potrebno.
4. Izvršiti program za dijete koje čita.
5. U roditelju zatvoriti kraj cjevovoda s kojeg se čita i obaviti druge pripreme ako je potrebno.
6. Ako drugo dijete treba pisati u cjevovod, stvoriti proces i izvršiti program.
7. Ako roditelj treba pisati, neka piše.

Čitanje i pisanje u cjevovod je slično radu s običnim datotekama, ali postoje i bitne razlike. Pisanje i čitanje se odvija po principu FIFO-redova: čitač čita redom kojim je pisač zapisivao. Ako čitač isprazni cjevovod on čeka podatke, a ako pisač popuni cjevovod on mora čekati dok ga čitač malo ne isprazni i oslobodi mjesto u cjevovodu. Svaki pročitani znak više ne postoji u cjevovodu i može se vratiti samo tako da ga pisač ponovo upiše.

Osnovni nedostatak komuniciranja preko cjevovoda je da procesi moraju biti povezani, na primjer roditelj i dijete. Cjevovod se ne može kreirati nakon što su procesi već stvoreni, zato što proces koji stvara cjevovod ne može prenjeti opisnik datoteke drugom procesu. Opisnici datoteka se prenose samo kod kreiranja procesa djeteta. Zato se prvo stvori cjevovod, a zatim kreira dijete koje će naslijediti opisnik datoteka cjevovoda. Procesi koji komuniciraju preko cjevovoda mogu biti roditelj i dijete, ili dvoje djece, ili "djed" i "unuk". Važno je samo da su u "srodstvu" i da je cjevovod prenesen kod "rođenja".

Cjevovodi koriste međuspremnike (*buffer cache*) veličine jednog bloka (obično 512 bajtova) kao i obične datoteke, što se može iskoristiti za povećanje efikasnosti rada. Svakim pozivom *write* moguće je upisati jedan blok podataka. Ako pisač ne piše kompletne blokove, a čitač pokušava čitati cijeli blok, čitač će dobivati nekompletne blokove. Ali ako je pisač brži od čitača onda će čitač ipak čitati kompletne blokove.

Korisnici mogu upotrebljavati cjevovode pozivima naredbi iz komandne linije, na primjer:

```
ls | wc
```

Tok podataka ima samo jedan smjer, od *ls* prema *wc*.

### Sustavski pozivi i funkcije za rad sa cjevovodima

Cjevovod je predstavljen s dva opisnika datoteka a kreira se sustavskim pozivom *pipe*:

```
int pipe(int fd[2]);
```

Ako je uspješno izvršen, *pipe* vraća 0, a u slučaju greške -1. *fd[1]* je deskriptor ulazne strane cjevovoda. Pisanjem u njega stavljaju se podaci u cjevovod, a čitanje iz *fd[0]* (deskriptor izlazne strane cjevovoda) vadi podatke van. Dobiveni deskriptori mogu se koristiti u pozivima za rad s datotekama: *close*, *dup*, *fcntl*, *fstat*, *read*, *write*. Izuzetak je *lseek* jer se cjevovodu može pristupati samo sekvencijalno (FIFO).

```
int close(int fd);
```

zatvara opisnik datoteke *fd*. Rezultat je 0, ili -1 u slučaju greške.

U radu s cjevovodima često je potrebno duplicirati postojeće opisnike datoteka (da bi se cjevovod povezao na standardni ulaz ili izlaz, da bi se opisnici za standardni ulaz i izlaz mogli spremiti i vratiti nakon zatvaranja cjevovoda, itd.). Za dupliciranje opisnika datoteka služi sustavski poziv *dup*:

```
int dup(int fd);
```

*dup* kopira postojeći opisnik datoteke *fd* i vraća novi opisnik datoteke ili -1 u slučaju greške. Poziv ne uspjeva, na primjer ako *fd* nije otvoren ili je već otvoren maksimalan broj opisnika datoteka (obično 20). Novi opisnik datoteke ima drugačiji broj od originalnog! Pravilo je da se kod otvaranja bilo kojeg novog opisnika datoteke uzima najmanji slobodni broj (to vrijedi i kod *open* i *pipe*). Najmanji brojevi 0, 1 i 2 su opisnici standardnog ulaza, standardnog izlaza i standardnog izlaza za greške.

Koristeći to pravilo, kraj cjevovoda iz kojeg se može čitati se povezuje kao standardni ulaz na slijedeći način: zatvori se opisnik datoteke 0 i duplicira se opisnik kraja za čitanje cjevovoda. *dup* će vratiti opisnik datoteke 0. Ako se zatim pokrene proces koji čita standardni ulaz, on će čitati iz cjevovoda. (Prethodno treba zatvoriti polazni opisnik kraja za čitanje cjevovoda ako ovaj više nije potreban.) Slično se postiže da opisnik datoteke 1 (standardni izlaz) bude kraj za pisanje u cjevovod. Na taj način ljuska povezuje procese koje korisnik poziva sa:

```
proc1 | proc2
```

Ako kasnije treba restaurirati opisnike datoteka za standardni ulaz i izlaz, onda ih je potrebno prvo duplicirati i zapamtiti tako dobivene opisnike. Tada se opisnici 0 i 1 mogu zatvoriti i zamijeniti, a kada ih je potrebno vratiti, primjenjuje se isti postupak za zamjenu opisnika. Na primjer, za restauraciju standardnog ulaza prvo se zatvara opisnik 0, a zatim se duplicira prije dobivena kopija standardnog ulaza. (Nakon toga se kopija može zatvoriti.)

Ponekad izbor najnižeg slobodnog broja za opisnik nove datoteke nije poželjan. Tada se umjesto *dup* može koristiti *fcntl* na slijedeći način:

```
int fcntl(fd, cmd, arg);
```

*fd* je deskriptor kojega se duplicira. *cmd* treba biti `F_DUPFD`, a odabrat će se novi deskriptor veći ili jednak *arg*.

BSD UNIX (ali ne i System V) ima još i poziv:

```
int dup2(int oldd, int newd);
```

Deskriptor *oldd* se duplicira u *newd*. Ako je *newd* već bio zauzet, on se prethodno zatvara. Rezultat je *newd* ili -1 u slučaju greške.

Čitanje i pisanje se provodi pozivima read i write kao i kod običnih datoteka:

```
int read(int fd, char *buf, unsigned nbyte);
```

pokušava pročitati *nbyte* znakova iz datoteke s opisnikom *fd* na adresu spremnika *buf*. Rezultat je broj stvarno pročitanih znakova (koji može biti manji od *nbyte*). U slučaju greške, rezultat je -1.

Ako je cjevovod prazan, *read* će čekati. Međutim, neće nužno čekati dok ne bude prisutno *nbytes* znakova nego će vratiti onoliko znakova koliko je prisutno u cjevovodu (ako ih ima manje od *nbytes*). *read* će vratiti znak kraja datoteke (rezultat 0) samo kada se zatvori opisnik kraja za pisanje u cjevovod.

```
int write(int fd, char *buf, unsigned nbyte);
```

pokušava zapisati *nbyte* znakova iz spremnika *buf* u datoteku *fd*. Rezultat je broj stvarno zapisanih znakova ili -1 u slučaju greške.

Ako se cjevovod napuni, *write* čeka dok se ne oslobodi prostor. *write* se neće djelomično obaviti već će čekati dok ne bude dovoljno mjesta za svih *nbyte* znakova. Kapacitet cjevovoda je tipično 5120 znakova (10 blokova). Ako se zatvori opisnik kraja za čitanje iz cjevovoda, *write* će završiti s greškom.

Pozivi *read* i *write* se rijetko upotrebljavaju izravno u programima jer C ima bogatu biblioteku praktičnijih funkcija za pristup datotekama (*fread*, *fwrite*, *printf*, *scanf*, itd.). Međutim, te funkcije ne rade sa opisnicima datoteka već sa kazaljkama na strukturu koja opisuje datoteku. Ta kazaljka se za obične datoteke dobiva funkcijom *fopen* (vidi `man fopen`), ali se može dobiti i iz opisnika datoteke funkcijom *fdopen*:

```
FILE *fdopen(int fd, char *type);
```

vraća kazaljku na strukturu koja opisuje datoteku za opisnik datoteke *fd*. U slučaju greške rezultat je `NULL`. *type* je niz znakova koji opisuje način pristupa datoteci i mora odgovarati načinu na koji je otvoren opisnik datoteke *fd*:

`"r"`čitanje

`"w"`pisanje

Ovo nije potrebno raditi ako se cjevovod povezuje na standardni ulaz ili izlaz jer uvijek postoje kazaljke *stdin*, *stdout* i *stderr* za standardni ulaz, standardni izlaz i standardni izlaz za poruke o greškama.

**Primjer korištenja cjevovoda**

Ovo je jednostavan primjer korištenja cjevovoda za prenošenje poruke iz jednog procesa u drugi. Proces prvo stvara cjevovod, a zatim jedno dijete. Dijete nasljeđuje sve podatke i opisnike, pa tako i cjevovod. Taj proces zatvara opisnik za pisanje u cjevovod i čita jednu poruku iz cjevovoda koju zatim ispisuje. S druge strane, roditelj, nakon kreiranja djeteta, zatvara opisnik za čitanje u cjevovod. Zatim šalje poruku cjevovodom i čeka da dijete završi.

Zbog jednostavnosti, uglavnom se ne provjerava je li došlo do grešaka kod pojedinih poziva. Upotrijebljeni opisnici cjevovoda se ne zatvaraju jer to ionako radi *exit*.

```
#include <stdio.h>
#include <string.h>
#define MAXREAD 20/* najveća duljina poruke*/
int main(void)
{
      int pfd[2];
      char buf[MAXREAD] = "";
      char message[] = "Kroz cijev!";/* poruka*/
      if (pipe(pfd) == -1)/* stvaranje cjevovoda*/
            exit(1);
      switch (fork()) {
            case -1: /* dijete nije kreirano*/
                  exit(1);
            case 0:/* dijete čita */
                  close(pfd[1]);/* zato zatvara kraj za
                  pisanje*/
                  (void) read(pfd[0], buf, MAXREAD);
                  puts(buf);
                  exit(0);
            default:/* roditelj piše */
                  close(pfd[0]);/* zato zatvara kraj za
                  čitanje*/
                  (void) write(pfd[1], message,
                  strlen(message) + 1);
                  wait(NULL);/* roditelj čeka da dijete
                  završi*/
            }
            exit(0);/* zatvara sve deskriptore */
      }
```

## IMENOVANI CJEVOVODI

Imenovani cjevovodi (FIFO redovi) su kombinacija datoteka i cjevovoda. Oni imaju svoje ime i mjesto u sustavu datoteka, ali su označeni kao posebna vrsta datoteka (oznaka `p` - *pipe*). Pristupa im se korištenjem imena, kao i datotekama, pa procesi ne moraju biti u srodstvu da bi komunicirali preko njih. Imenovani cjevovod je potrebno prvo kreirati sustavskim pozivom mknod, a zatim ga treba otvoriti dva puta: jednom za čitanje, a jednom za pisanje. Ovisno o načinu pristupa navedenom kod otvaranja datoteke, proces otvara kraj za čitanje ili kraj za pisanje imenovanog cjevovoda. Nakon što su otvoreni s njima se radi kao s cjevovodima. Kapacitet imenovanih cjevovoda ovisi o implementaciji.

Kad se imenovani cjevovod otvara za čitanje, *open* čeka dok ga neki drugi proces ne otvori i za pisanje. Vrijedi i obrnuto. To dozvoljava procesima da se sinkroniziraju prije nego počne prenošenje bilo kakvih podataka.

S druge strane, kada je u *open* za čitanje postavljena zastavica `O_NDELAY`, on neće čekati odgovarajući *open* za pisanje, a kada je postavljena u *open* za pisanje, on će vratiti grešku ako niti jedan čitač nema otvoren isti imenovani cjevovod. Svrha ovoga je da procesi ne pišu u cjevovode koje u tom trenutku nitko ne čita zato što UNIX ne sprema podatke u njih trajno. Ako u imenovanom cjevovodu ostanu podaci nakon zatvaranja svih opisnika datoteka koji su s njim povezani, bit će izgubljeni bez dojave greške.

Ako zastavica `O_NDELAY` nije postavljena, *read* se zablokira kada nema podataka u imenovanom cjevovodu, a *write* se zablokira ako je prepunjen. Ako je zastavica postavljena, ni r*ead* ni *write* se ne zablokiraju nego jave grešku.

**Sustavski pozivi za rad sa imenovanim cjevovodima**

```
int mknod(char *path, int mode, int dev);
```

stvara novu datoteku čiji put (uključujući i ime) je *path*. Ovaj poziv je rezerviran za superkorisnika, osim u slučaju stvaranja imenovanog cjevovoda kada *mode* mora biti kombinacija zastavice `S_IFIFO` (oktalno 0010000, definirano u `<sys/stat.h>`) i dozvola pristupa u donjih 9 bitova, a *dev* nije bitno). Rezultat je 0, osim u slučaju greške kada je -1.

Korisnik može stvoriti imenovani cjevovod naredbom:

```
/etc/mknod ime p
```

```
int open(char *path, int flags [, int mode] );
```

*path* pokazuje na put (s imenom) datoteke. *flags* je neka kombinacija zastavica definiranih u `<fcntl.h>`:

`O_RDONLY` otvori za čitanje;

`O_WRONLY` otvori za pisanje;

`O_RDWR` otvori za čitanje i pisanje;

`O_NDELAY` utječe na postupak otvaranja i kasniji rad sa *read* i *write* - ako se operacija ne može obaviti bez čekanja, nema čekanja nego se vraća greška;

`O_APPEND` dodavanje na kraj datoteke;

`O_SYNC` write će čekati sve dok podaci ne budu stvarno zapisani na disk;

`O_CREAT` ako datoteke nema, kreira se nova sa pravima pristupa danim u *mode*;

`O_TRUNC` ako datoteka postoji, njen sadržaj se briše;

`O_EXCL` ne dozvoljava korištenje postojeće datoteke ako je postavljeno `O_CREAT` .

Ako je datoteka (ili imenovani cjevovod) uspješno otvorena, *open* vraća opisnik datoteke (uvijek najmanji koji može). U slučaju greške, rezultat je -1.

Neke od zastavica postavljenih kod otvaranja, mogu se u kasnijem radu mijenjati korištenjem sustavskog poziva *fcntl* (vidi `man fcntl`):

```
int fcntl(int fd, int cmd, int arg);
```

*fd* je opisnik datoteke, *cmd* treba biti `F_SETFL`, a *arg* kombinacija zastavica koja se postavlja. Smatra se da je poziv uspio ako je rezultat različit od -1, ali se samo neke zastavice mogu mijenjati (na primjer, `O_NDELAY`).

Ako je *cmd* jednak `F_GETFL`, ovim pozivom se kao rezultat dobiva stanje svih zastavica za otvorenu datoteku s opisnikom *fd*.

Dalji rad sa imenovanim cjevovodima je isti kao i sa običnim cjevovodima. Koriste se pozivi *read* i *write* ili funkcije iz standardne biblioteke koje unutar sebe također koriste ove pozive za pristup podacima.

Primjer programa s imenovanim cjevovodima